

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Клименко Алёна Сергеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.0.1	Отладка программ с помощью GDB	13
4.0.2	Добавление точек остановок	18
4.0.3	Работа с данными программы в GDB	20
4.0.4	Обработка аргументов командной строки в GDB	25
4.1	Задание для самостоятельной работы	26
5	Выводы	32
	Список литературы	33

Список иллюстраций

4.1	Создание рабочего каталога	8
4.2	файл с кодом	9
4.3	Запуск программы из листинга	10
4.4	Изменение программы первого листинга	11
4.5	работа кода	13
4.6	Запуск программы в отладчике	14
4.7	Проверка программы отладчиком	15
4.8	Запуск отладчика с брейкпойнтом	16
4.9	Дисассимилирование программы	17
4.10	Режим псевдографики	18
4.11	Список брейкпойнтов	19
4.12	Добавление второй точки останова	20
4.13	Просмотр содержимого регистров	21
4.14	Просмотр содержимого переменных двумя способами	22
4.15	Изменение содержимого переменных двумя способами	23
4.16	Просмотр значения регистра разными представлениями	24
4.17	Примеры использования команды set	25
4.18	Подготовка новой программы	25
4.19	Проверка работы стека	26
4.20	Измененная программа предыдущей лабораторной работы	27
4.21	Поиск ошибки в программе через пошаговую отладку	29
4.22	Проверка корректировок в программе	30

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

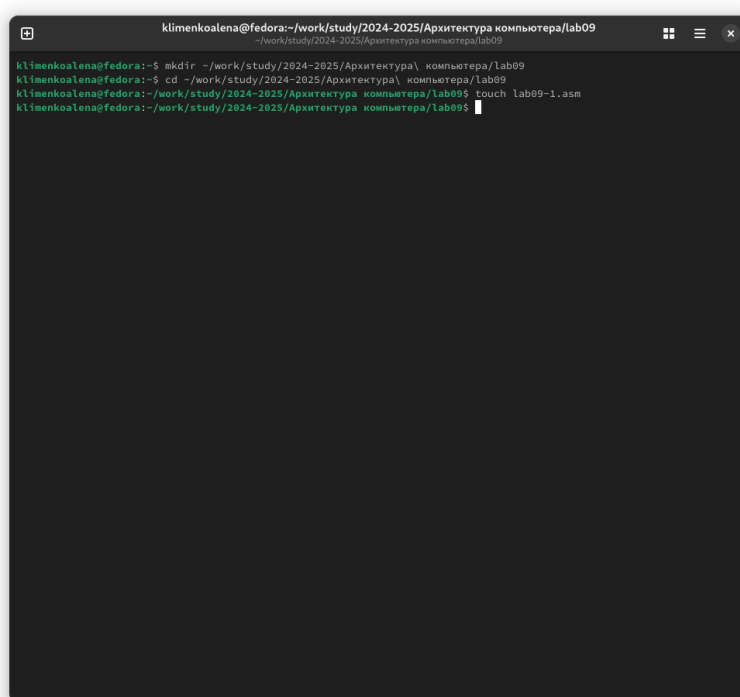
- синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

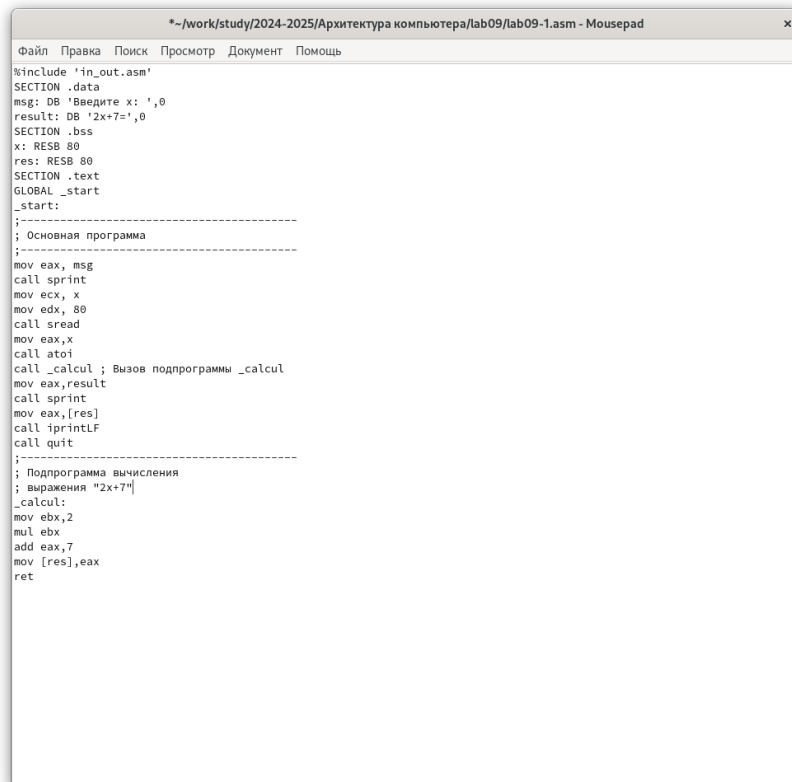
создаю каталог для выполнения лабораторной работы №9 (рис. 4.1).



```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~$ mkdir ~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~$ cd ~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ touch lab09-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$
```

Рис. 4.1: Создание рабочего каталога

Копирую в файл код из листинга (рис. 4.2).



```

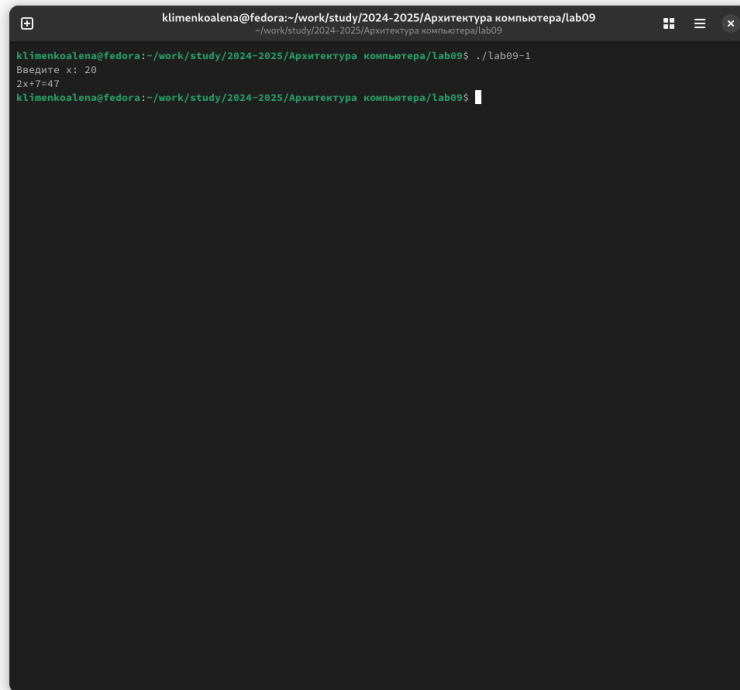
*~/work/study/2024-2025/Архитектура компьютера/lab09/lab09-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret

```

Рис. 4.2: файл с кодом

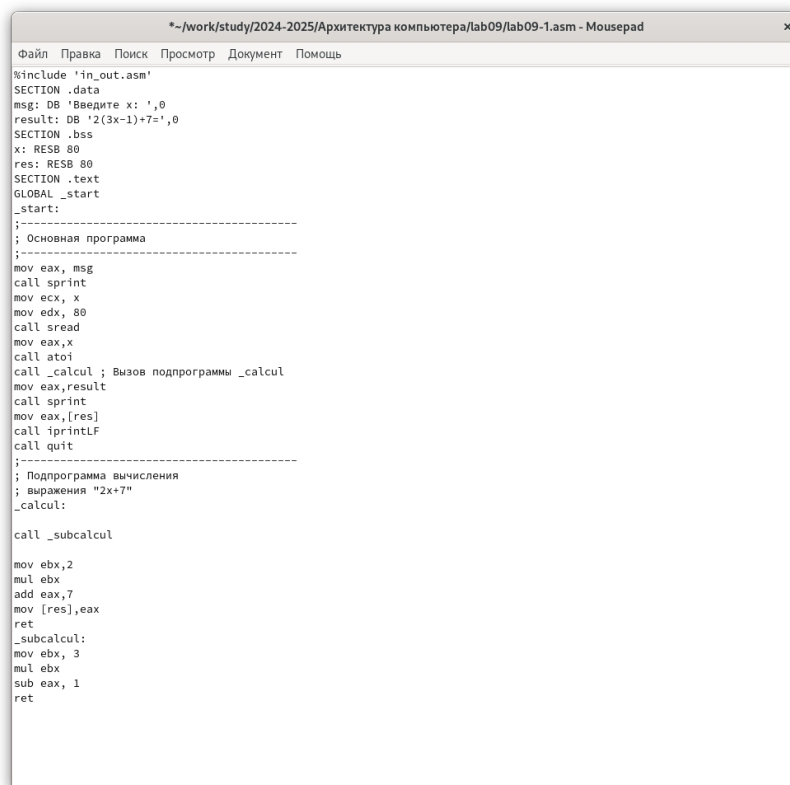
компилирую и запускаю его, данная программа выполняет вычисление функции(рис. 4.3).

A terminal window with a dark background. The title bar shows the user 'klimenkoalena' on a 'fedora' machine, in the directory '~/work/study/2024-2025/Архитектура компьютера/lab09'. The terminal content shows the user running './lab09-1', which prompts 'Введите x: 20'. The user enters '20', and the program outputs '2x+7=47'. The prompt returns to 'klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09\$' with a cursor.

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ./lab09-1
Введите x: 20
2x+7=47
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$
```

Рис. 4.3: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму (рис. 4.4).



```
*~/work/study/2024-2025/Архитектура компьютера/lab09/lab09-1.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul

mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 4.4: Изменение программы первого листинга

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ', 0
result: DB '2(3x-1)+7=', 0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
```

```

_start:
mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

call _calcul

mov eax, result
call sprint
mov eax, [res]
call iprintLF

call quit

_calcul:
push eax
call _subcalcul

mov ebx, 2
mul ebx
add eax, 7

mov [res], eax

```

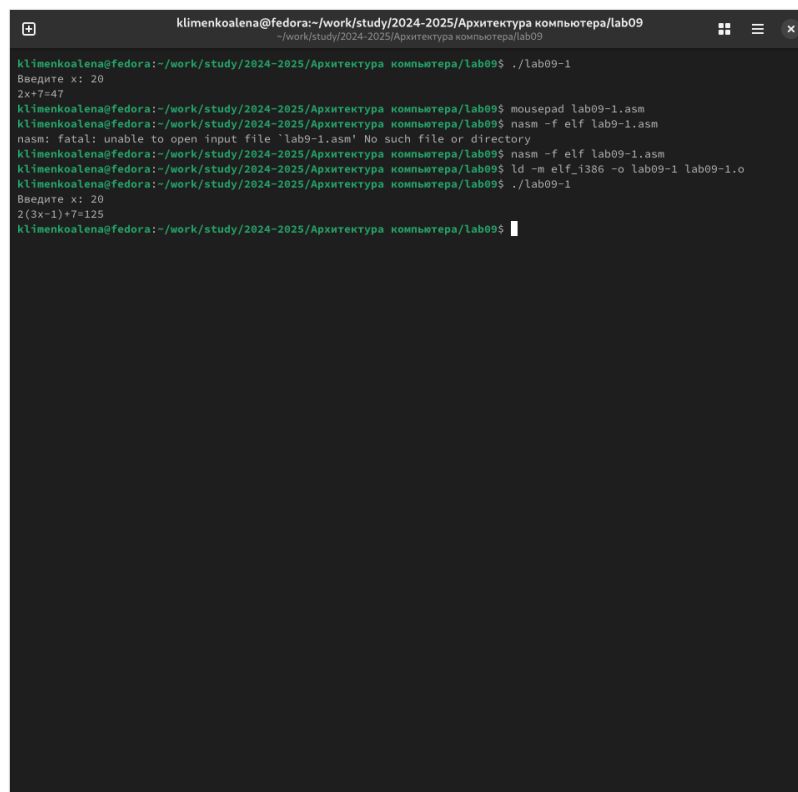
```

pop eax
ret

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret

```

теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 4.5).



```

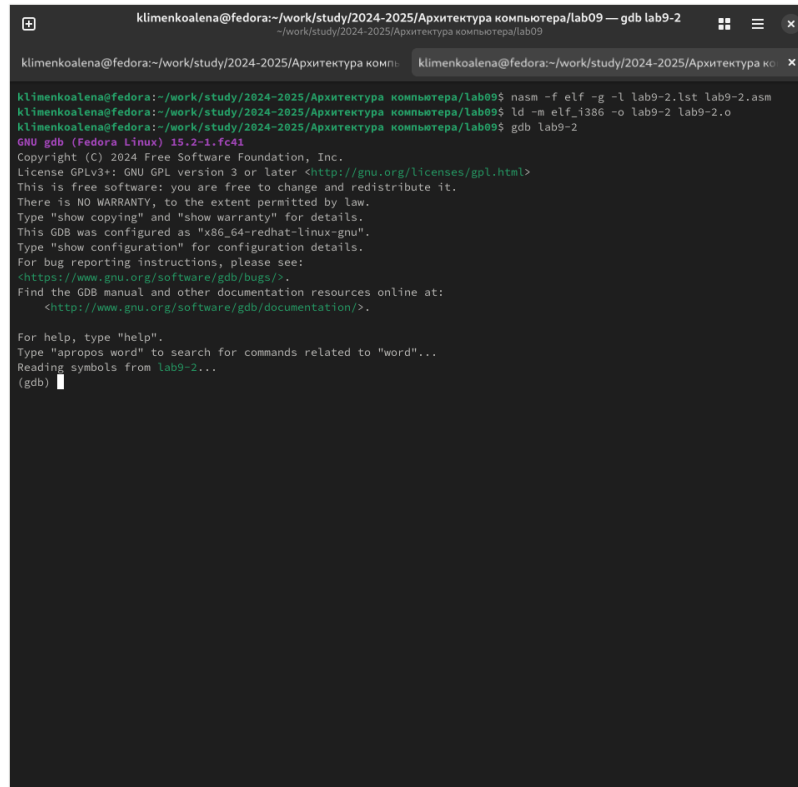
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ./lab09-1
Введите x: 20
2x+7=47
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ mousepad lab09-1.asm
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ nasm -f elf lab09-1.asm
nasm: fatal: unable to open input file 'lab09-1.asm' No such file or directory
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ nasm -f elf lab09-1.asm
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ./lab09-1
Введите x: 20
2(3x-1)+7=125
klimenko@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$

```

Рис. 4.5: работа кода

4.0.1 Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 4.6).



```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура к:
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) █
```

Рис. 4.6: Запуск программы в отладчике

Запустив программу командой `gdb`, я убедился в том, что она работает исправно (рис. 4.7).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура к:
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading 47.72 K separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4360) exited normally]
(gdb)
```

Рис. 4.7: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку `_start` и снова запускаю отладку (рис. 4.8).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура к:
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading 47.72 K separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4360) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) 
```

Рис. 4.8: Запуск отладчика с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel *амд топчик* (рис. 4.9).

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как *ax*, *eax*, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).


```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс: x

Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading 47.72 K separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4360) exited normally]
(gdb) break _start
Breakpoint 1 at 0x0049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x0049000 <+0>: mov $0x4,%eax
0x0049005 <+5>: mov $0x1,%ebx
0x004900a <+10>: mov $0x04a000,%ecx
0x004900f <+15>: mov $0x8,%edx
0x0049014 <+20>: int $0x80
0x0049016 <+22>: mov $0x4,%eax
0x004901b <+27>: mov $0x1,%ebx
0x0049020 <+32>: mov $0x04a000,%ecx
0x0049025 <+37>: mov $0x7,%edx
0x004902a <+42>: int $0x80
0x004902c <+44>: mov $0x1,%eax
0x0049031 <+49>: mov $0x0,%ebx
0x0049036 <+54>: int $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x0049000 <+0>: mov eax,0x4
0x0049005 <+5>: mov ebx,0x1
0x004900a <+10>: mov ecx,0x04a000
0x004900f <+15>: mov edx,0x8
0x0049014 <+20>: int 0x80
0x0049016 <+22>: mov eax,0x4
0x004901b <+27>: mov ebx,0x1
0x0049020 <+32>: mov ecx,0x04a000
0x0049025 <+37>: mov edx,0x7
0x004902a <+42>: int 0x80
0x004902c <+44>: mov eax,0x1
0x0049031 <+49>: mov ebx,0x0
0x0049036 <+54>: int 0x80
End of assembler dump.
(gdb) |
```

Рис. 4.9: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 4.10).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура к: x

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcee0  0xffffcee0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

B> 0x8049000 <_start> mov    eax,0x4
0x8049005 <_start+5> mov    ebx,0x1
0x804900a <_start+10> mov    ecx,0x804a000
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int    0x80
0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a000
0x8049025 <_start+37> mov    edx,0x7
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
0x8049031 <_start+49> mov    ebx,0x8
0x8049036 <_start+54> int    0x80
0x8049038 add     BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) 
```

Рис. 4.10: Режим псевдографики

4.0.2 Добавление точек остановок

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 4.11).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс: x

Register group: general
eax    0x0      0      ecx    0x0      0
edx    0x0      0      ebx    0x0      0
esp    0xffffcee0 0xffffcee0 ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x8049000 0x8049000 <_start> eflags 0x202    [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

B> 0x8049000 <_start> mov    eax,0x4
0x8049005 <_start+5> mov    ebx,0x1
0x804900a <_start+10> mov    ecx,0x804a000
0x804900f <_start+15> mov    edx,0x8
0x8049014 <_start+20> int    0x80
0x8049016 <_start+22> mov    eax,0x4
0x804901b <_start+27> mov    ebx,0x1
0x8049020 <_start+32> mov    ecx,0x804a000
0x8049025 <_start+37> mov    edx,0x7
0x804902a <_start+42> int    0x80
0x804902c <_start+44> mov    eax,0x1
0x8049031 <_start+49> mov    ebx,0x8
0x8049036 <_start+54> int    0x80
0x8049038 add    BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint       keep y   0x8049000 lab9-2.asm:9
       breakpoint already hit 1 time
(gdb) break *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab9-2.asm, line 9.
(gdb) t b
Num    Type             Disp Enb Address      What
1      breakpoint       keep y   0x8049000 lab9-2.asm:9
       breakpoint already hit 1 time
2      breakpoint       keep y   0x8049000 lab9-2.asm:9
(gdb) 
```

Рис. 4.11: Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. 4.12).

The screenshot shows a GDB terminal window with the title bar "klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура ко...". The window is divided into two main sections. The top section, titled "Registers", displays "[Register Values Unavailable]". The bottom section shows assembly code with a red box highlighting a specific block of instructions:

```
b+ 0x8049000 <_start>    mov    eax,0x4
0x8049005 <_start+5>    mov    ebx,0x1
0x804900a <_start+10>   mov    ecx,0x804a000
0x804900f <_start+15>   mov    edx,0x8
b+ 0x8049014 <_start+20> int    0x80
0x8049016 <_start+22>   mov    eax,0x4
```

Below the assembly code, the terminal shows the execution of several GDB commands:

```
exec No process (asm) In: L?? PC: ??
(gdb) break *0x8049000
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y   0x08049000 lab9-2.asm:9
(gdb) b *0x8049014
Breakpoint 2 at 0x8049014: file lab9-2.asm, line 13.
(gdb)
```

Рис. 4.12: Добавление второй точки останова

4.0.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` (рис. 4.13).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс: x

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcee0  0xffffcee0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

0x80490d2  add  BYTE PTR [eax],al
0x80490d4  add  BYTE PTR [eax],al
0x80490d6  add  BYTE PTR [eax],al
0x80490d8  add  BYTE PTR [eax],al
0x80490da  add  BYTE PTR [eax],al
0x80490dc  add  BYTE PTR [eax],al
0x80490de  add  BYTE PTR [eax],al
0x80490e0  add  BYTE PTR [eax],al
0x80490e2  add  BYTE PTR [eax],al
0x80490e4  add  BYTE PTR [eax],al
0x80490e6  add  BYTE PTR [eax],al
0x80490e8  add  BYTE PTR [eax],al
0x80490ea  add  BYTE PTR [eax],al
0x80490ec  add  BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcee0  0xffffcee0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000  0x8049000 <_start>
eflags    0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.13: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 4.14).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс:

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcee0  0xffffcee0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

0x80490d2 add BYTE PTR [eax],al
0x80490d4 add BYTE PTR [eax],al
0x80490d6 add BYTE PTR [eax],al
0x80490d8 add BYTE PTR [eax],al
0x80490da add BYTE PTR [eax],al
0x80490dc add BYTE PTR [eax],al
0x80490de add BYTE PTR [eax],al
0x80490e0 add BYTE PTR [eax],al
0x80490e2 add BYTE PTR [eax],al
0x80490e4 add BYTE PTR [eax],al
0x80490e6 add BYTE PTR [eax],al
0x80490e8 add BYTE PTR [eax],al
0x80490ea add BYTE PTR [eax],al
0x80490ec add BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
esi      0x0      0
edi      0x0      0
eip      0x8049000  0x8049000 <_start>
eflags    0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a006 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

Рис. 4.14: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. 4.15).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс: x

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcee0  0xffffcee0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000 <_start>  eflags  0x202      [ IF ]
cs       0x23      35      ss       0x2b      43
ds       0x2b      43      es       0x2b      43
fs       0x0      0      gs       0x0      0

0x80490d2  add  BYTE PTR [eax],al
0x80490d4  add  BYTE PTR [eax],al
0x80490d6  add  BYTE PTR [eax],al
0x80490d8  add  BYTE PTR [eax],al
0x80490da  add  BYTE PTR [eax],al
0x80490dc  add  BYTE PTR [eax],al
0x80490de  add  BYTE PTR [eax],al
0x80490e0  add  BYTE PTR [eax],al
0x80490e2  add  BYTE PTR [eax],al
0x80490e4  add  BYTE PTR [eax],al
0x80490e6  add  BYTE PTR [eax],al
0x80490e8  add  BYTE PTR [eax],al
0x80490ea  add  BYTE PTR [eax],al
0x80490ec  add  BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
es      0x2b      43
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set (char)&msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set (char)&msg1=h
No symbol "h" in current context.
(gdb) set (char)&msg1 = 'h'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "world!\n\034"
(gdb) 
```

Рис. 4.15: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. 4.16).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/lab09 — gdb lab9-2
~/work/study/2024-2025/Архитектура компьютера/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура комп: klimenkoalena@fedora:~/work/study/2024-2025/Архитектура кс: x

Register group: general
eax    0x34      52      ecx    0x31      49
edx    0x0       0       ebx    0x32      50
esp    0xffffcee0 0xffffcee0 ebp    0x0       0x0
esi    0x0       0       edi    0x0       0
eip    0x8049000 0x8049000 <_start> eflags 0x202      [ IF ]
cs     0x23      35      ss     0x2b      43
ds     0x2b      43      es     0x2b      43
fs     0x0       0       gs     0x0       0

0x80490d2 add BYTE PTR [eax],al
0x80490d4 add BYTE PTR [eax],al
0x80490d6 add BYTE PTR [eax],al
0x80490d8 add BYTE PTR [eax],al
0x80490da add BYTE PTR [eax],al
0x80490dc add BYTE PTR [eax],al
0x80490de add BYTE PTR [eax],al
0x80490e0 add BYTE PTR [eax],al
0x80490e2 add BYTE PTR [eax],al
0x80490e4 add BYTE PTR [eax],al
0x80490e6 add BYTE PTR [eax],al
0x80490e8 add BYTE PTR [eax],al
0x80490ea add BYTE PTR [eax],al
0x80490ec add BYTE PTR [eax],al

native process 4383 (asm) In: _start L9 PC: 0x8049000
$1 = 0
(gdb) set $ebx = '2'
(gdb) p/s $ebx
$2 = 50
(gdb) set $eax = '4'
(gdb) p/s $eax
$3 = 52
(gdb) p/t $eax
$4 = 110100
(gdb) p/s $ecx
$5 = 0
(gdb) set $ecx = '1'
(gdb) p/x $ecx
$6 = 0x31
(gdb)
```

Рис. 4.16: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистров (рис. 4.17).

The screenshot shows the GDB interface with the following content:

Register group: general

Register	Value	Register	Value
eax	0x34	ecx	0x31
edx	0x0	ebx	0x32
esp	0xffffcee0	ebp	0x0
esi	0x0	edi	0x0
eip	0x8049000	eflags	0x202
cs	0x23	ss	0x2b
ds	0x2b	es	0x2b
fs	0x0	gs	0x0

Assembly code (0x80490d2 to 0x80490ec):

```
0x80490d2 add BYTE PTR [eax],al
0x80490d4 add BYTE PTR [eax],al
0x80490d6 add BYTE PTR [eax],al
0x80490d8 add BYTE PTR [eax],al
0x80490da add BYTE PTR [eax],al
0x80490dc add BYTE PTR [eax],al
0x80490de add BYTE PTR [eax],al
0x80490e0 add BYTE PTR [eax],al
0x80490e2 add BYTE PTR [eax],al
0x80490e4 add BYTE PTR [eax],al
0x80490e6 add BYTE PTR [eax],al
0x80490e8 add BYTE PTR [eax],al
0x80490ea add BYTE PTR [eax],al
0x80490ec add BYTE PTR [eax],al
```

GDB Session Log:

```
native process 4383 (asm) In: _start
$1 = 0
(gdb) set $ebx = '2'
(gdb) p/s $ebx
$2 = 50
(gdb) set $eax = '4'
(gdb) p/s $eax
$3 = 52
(gdb) p/t $eax
$4 = 110100
(gdb) p/s $ecx
$5 = 0
(gdb) set $ecx = '1'
(gdb) p/x $ecx
$6 = 0x31
(gdb)
```

Рис. 4.17: Примеры использования команды set

4.0.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. 4.18).

```
klimenkoalena@fedora:~$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
```

Рис. 4.18: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились. (рис. 4.19).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb --args lab9...
~work/study/2024-2025/Архитектура компьютера/arch-pc/lab09

klimenkoalena@fedora:~/work/: klimenkoalena@fedora:~/work/: klimenkoalena@fedora:~/work/: klimenkoalena@fedora:~/wo

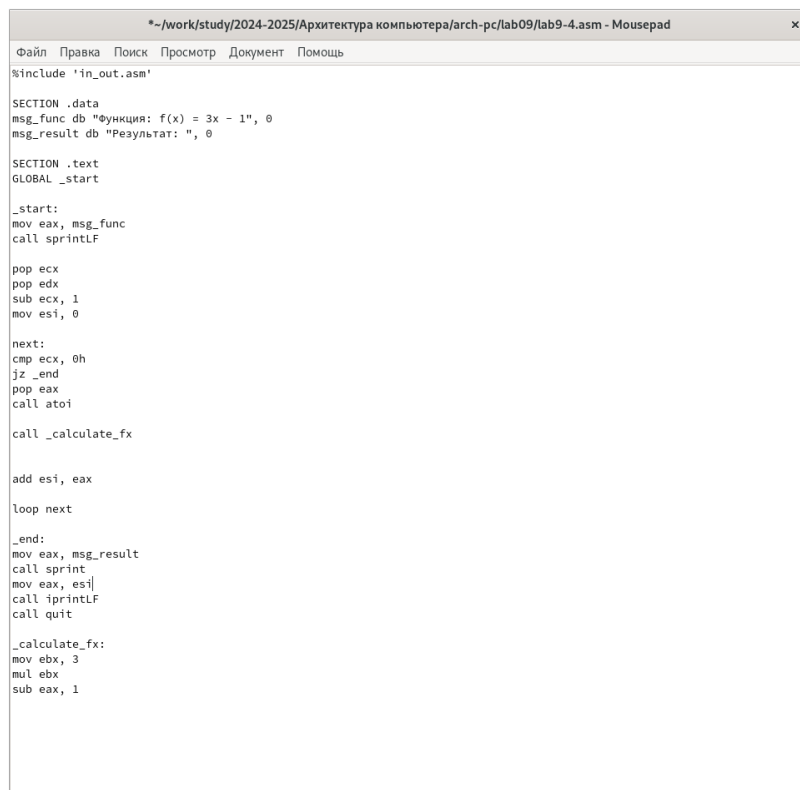
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:7
7       pop ecx
(gdb) x/x $esp
0xffffd000: 0x00000005
(gdb) x/s *(void**)(($esp + 4))
0xffffd00b: "/home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(($esp + 4))
0xffffd00b: "/home/klimenkoalena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(($esp + 8))
0xffffd00d: "arg1"
(gdb) (gdb) x/s *(void**)(($esp + 12))
Undefined command: "". Try "help".
(gdb) x/s *(void**)(($esp + 12))
0xffffd00d: "arg"
(gdb) x/s *(void**)(($esp + 16))
0xffffd00d: "2"
(gdb) x/s *(void**)(($esp + 20))
0xffffd00d: "arg 3"
(gdb) x/s *(void**)(($esp + 24))
0x0: 
```

Рис. 4.19: Проверка работы стека

4.1 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 4.20).



```
*~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 3x - 1", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

_calculate_fx:
mov ebx, 3
mul ebx
sub eax, 1
```

Рис. 4.20: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf
```

```

pop ecx
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4

```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю

изменение значений регистров через `i` г. При выполнении инструкции `mul` `ecx` можно заметить, что результат умножения записывается в регистр `eax`, но также меняет и `edx`. Значение регистра `ebx` не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. 4.21).

```

klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-5
~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09

Register group: general
eax 0x2 2 ecx 0x4 4
edx 0x0 0 ebx 0x5 5
esp 0xffffced0 0xffffced0 ebp 0x0 0
esi 0x0 0 edi 0x0 0
eip 0x80490f9 0x80490f9 <_start+17> eflags 0x206 [ PF IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

B+ 0x80490e8 <_start> mov $0x1,%ebx
0x80490ed <_start+5> mov $0x2,%ecx
0x80490f2 <_start+10> add %eax,%ebx
0x80490f4 <_start+12> mov $0x4,%ecx
> 0x80490f9 <_start+17> mul %ecx
0x80490fb <_start+19> add $0x1,%ebx
0x80490fe <_start+22> mov %ebx,%edi
0x8049100 <_start+24> mov $0x804a000,%eax
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mov %edi,%eax
0x804910c <_start+36> call 0x8049006 <printf>
0x8049111 <_start+41> call 0x804900b <quit>
0x8049116 add %al,(%eax)
0x8049118 add %al,(%eax)
0x804911a add %al,(%eax)

native process 5965 (asm) In: _start L12 PC: 0x80490f9
eax 0x2 2
ecx 0x4 4
edx 0x0 0
ebx 0x5 5
esp 0xffffced0 0xffffced0
ebp 0x0 0
esi 0x0 0
edi 0x0 0
eip 0x80490f9 0x80490f9 <_start+17>
eflags 0x206 [ PF IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.21: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 4.22).

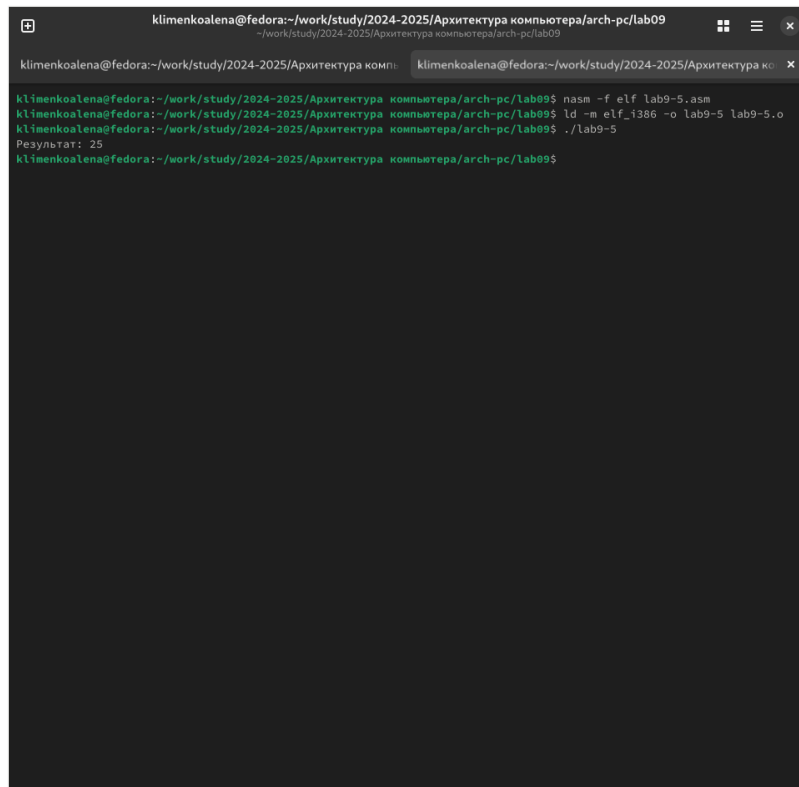


Рис. 4.22: Проверка корректировок в программе

Код измененной программы:

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0

SECTION .text
GLOBAL _start
_start:

mov ebx, 3
mov eax, 2
add ebx, eax
```

```
mov eax, ebx
mov ecx, 4
mul ecx
add eax, 5
mov edi, eax
```

```
mov eax, div
call sprint
mov eax, edi
call iprintLF
```

```
call quit
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №9